USING A PUBLIC BINARY SEMAPHORE FOR TASK SYNCHRONIZATION

The typical use case has one task (the "Taker") in some memory space (RTP or the kernel) that will pend upon a binary semaphore, and another task (the "Giver") in some other memory space that will give the semaphore, thereby unblocking or synchronizing the Taker.

TASK GUIDE

Step	Action	Example
1	Create a project for each participant in the sharing of the semaphore.	

2	For the Taker, the first task to execute, write code that does the following:
	
	a) Creates the semaphore and captures its ID
	b) Loops, pending on a semaphore, printing a message when it is given
		The code will look something like this:
		`SEM_ID semId;`
		`semId = semOpen("/binSem", SEM_TYPE_BINARY,`
		`SEM_EMPTY, SEM_Q_FIFO,`
		`OM_CREATE, NULL);`
		`.`
		`.`
		`.`
		`FOREVER{`
		`semTake(semId, WAIT_FOREVER);`
		`printf ("Got the semaphore\n");`
		`}`

3	For the Giver, write code that does the following:
	
	a) Attaches to (gains access to) the queue and captures its ID
	b) Loops, sending a message every 5 seconds, for example
		The code will look something like this:
		`SEM_ID semId;`
		`semId = semOpen("/binSem", 0,0,0,0,0);`
		`.`
		`.`
		`.`
		`FOREVER{`
		`semGive(semId);`
		`taskDelay (sysClkRateGet() * 5);`
		`}`

Key Points

- The semOpen () call is used either to create the public task initially or to gain access to it. The only difference is the OM_CREATE option.
- The basic dialog could also be extended to have multiple Givers (very common) or Takers (less so). That extension is not covered here.